A Semi-Autonomic Framework for Intrusion Tolerance in Heterogeneous Networks

Salvatore D'Antonio[1], Simon Pietro Romano[2], Steven Simpson[3], Paul Smith[3], and David Hutchison[3]

[1] CINI – ITeM Laboratory
Via Cinthia 80126 Napoli, Italy
[2] University of Napoli "Federico II"
Via Claudio 21 – 80125 Napoli, Italy
{saldanto, spromano}@unina.it
[3] Computing Department, InfoLab21
Lancaster University, Lancaster, UK
{ss, p.smith, dh}@comp.lancs.ac.uk

Abstract. A suitable strategy for network intrusion tolerance—detecting intrusions and remedying them—depends on aspects of the domain being protected, such as the kinds of intrusion faced, the resources available for monitoring and remediation, and the level at which automated remediation can be carried out. The decision to remediate autonomically will have to consider the relative costs of performing a potentially disruptive remedy in the wrong circumstances and leaving it up to a slow, but more accurate, human operator. Autonomic remediation also needs to be withdrawn at some point – a phase of recovery to the normal network state.
In this paper, we present a framework for deploying domain-adaptable intrusion-tolerance strategies in heterogeneous networks. Functionality is divided into that which is fixed by the domain and that which should adapt, in order to cope with heterogeneity. The interactions between detection and remediation are considered in order to make a stable recovery decision. We also present a model for combining diverse sources of monitoring to improve accurate decision making, an important pre-requisite to automated remediation.

1 Introduction

Network intrusion tolerance—detecting intrusions and remedying them—can be carried out manually or automatically, with various trade-offs of time and reliability. The choice is influenced by the resources available to the organization responsible for protecting a network. The kinds of attacks faced and the actual detection and remediation mechanisms may also vary depending the kind of network to be protected.

Automating intrusion tolerance could be problematic if it is applied in the wrong circumstances (e.g., a node is isolated because it is incorrectly supposed to be compromised). Also, when a temporary remedy is applied automatically, it

may affect the original detection of the intrusion, and one must consider whether to use that original detection mechanism to also detect the end of the intrusion and withdraw the remedy automatically.

As part of the INTERSECTION project [1], we have devised a framework for deploying security and resilience strategies against intrusions in networks. Here, we discuss its design relating to the problems of heterogeneity, and the automation of both deployment and withdrawal of mitigation mechanisms.

This section continues by discussing the problems of automating remediation to achieve intrusion tolerance, and how to achieve it in heterogeneous environments. Section 2 surveys intrusion-detection systems and touches on some systems that enable automatic remediation. Section 3 presents a framework for deploying intrusion-tolerant systems. Section 4 describes how a part of the framework is to be implemented to improve confidence of the initial detection.

1.1 To automate intrusion tolerance

Automation of network intrusion tolerance is desirable because of the unpredictability of attacks and the time taken by human operators to respond manually, which could otherwise lead to significant loss of service and financial cost. However, an automated intrusion-tolerance system that overreacts to an anomalous but innocent event can also be costly, so a balance must be struck between fast automation and more accurate but slow manual control.

The decision to perform different forms of remediation automatically will involve a trade-off between the potential impact on the threatened service if detection is mistaken and the cost of human intervention. An intensive attack will require an immediate response, even if it's not an ideal solution (involving some cost of its own), as it may take several hours for an operator to come up with a better solution. The cost of doing nothing in the meantime outweighs the cost of doing the automatic response.

This balance may vary across domains. Network-based malicious behaviour is becoming increasingly profit-driven, as attacks have shifted from being mostly targeted at large governmental and commercial organisations towards more profit-yielding small-to-medium enterprises (SMEs), and individuals. Larger organisations can afford 24-hour staffing, thereby reducing the need for automation and the risks of false positives. This is more difficult for SMEs, who may prefer to risk downtime for false positives if they can automatically recover quickly too.

1.2 Intrusion tolerance in heterogeneous environments

The hardware and software resources available for intrusion tolerance may vary significantly. For example, on a wireless mesh network (WMN) [17], there may be little in the way of resources, and those available may be highly constrained mesh devices; whereas in an enterprise setting, dedicated hardware may be available. Related to this point are the probable attacks a domain may face. For example, an end-system on a community WMN is unlikely to be the victim of a TCP

SYN attack (as the most likely victims, servers, are better placed on a wired network), which is distinct from the servers of a large financial institution.

This situation suggests that the mechanisms available for monitoring network traffic with the aim of detecting attacks, and the strategies for remedying them are very much specific to a domain. While the attacks that can occur within a domain are likely to be domain-specific, they will be drawn from a set of known attacks or attack types. This suggests that re-usable approaches to detecting attacks and general strategies for dealing with them (that can use locally-relevant monitoring and remediation mechanisms) can be developed.

2 Related Work

Intrusion Detection Systems (IDSs) can be classified as belonging to two main groups, depending on the detection technique employed: *anomaly detection* and *misuse detection*, also known as *signature detection* [5]. Both techniques depend on the existence of a reliable characterization of what is *normal* and what is not, in a particular networking scenario. Anomaly detection techniques base their evaluations on a model of what is normal, and classify as anomalous all the events that fall outside such a model. Indeed, if anomalous behaviour is recognized, this does not necessarily imply that an attack activity has occurred. Thus, a serious problem exists with anomaly detection techniques which generate a great amount of false alarms. Conversely, the primary advantage of anomaly detection is its ability to discover novel attacks.

An example anomaly detection system is presented in [10], where the authors propose a methodology to detect and classify network anomalies by means of analysis of traffic feature distributions; they adopt *entropy* as a metric to capture the degree of dispersal or concentration of the computed distributions. NETAD [13] detects anomalies based on analysis of packet structure: the system flags suspicious packets based on unusual byte values in network traffic.

The most known open-source signature-based intrusion detection systems are SNORT [6] and BRO [14]. These systems allow the user to define a customized set of rules in order to codify specific types of attacks. P-BEST [12] is a signature-based intrusion detection system able to detect computer and network misuse by means of a rule translator, a library of run-time routines, and a set of garbage collection routines.

There are approaches that aim to ensure network security by exploiting traffic monitoring information. In [2] and [16], the authors describe how to correlate netflow system and network views for intrusion detection. Their approach is human-driven, since they propose to use visualization tools in order to obtain useful information for security purposes. This approach demonstrates how data collected by flow monitoring systems can be used in the context of intrusion detection. In [3] and [11], data coming from both network monitoring and system logs are correlated in order to detect potential attacks. The authors prove that using data from more sources increases IDS performance. However, system logs are not always available, as in the case of servers owned by Internet providers.

Automatic remediation can cause problems if not applied to genuine intrusions. In [7], the authors highlight how automated intrusion response is often disabled due to the cost of responding to too many false positives. Their solution is to balance the cost of restarting more components of a system against the cost of not restarting enough, to produce an optimum response strategy even with uncertainty about the intrusion. In [4], an automatic approach to mitigating the effects of large volumes of ARP traffic caused by the scanning behaviour of Internet worms on switched networks is presented. ARP requests are dropped probabilistically in proportion to rate, and this is effective against supposedly compromised systems which generate the higher rates. However, it is shown the network's gateway router is unfairly affected, as it *legitimately* generates many ARP requests too.

3 The INTERSECTION Framework

The main goal of the INTERSECTION project [1] is to provide an infrastructure for heterogeneous networks to be resilient and secure in the face of intrusions, and to interoperate to achieve that resilience and security. We now present a framework for deploying intrusion-tolerance strategies that can adapt to heterogeneous domains, and is thus able to meet that goal.

The functions of our framework, as shown in Fig. 1, form a control loop that enables automatic intrusion tolerance, and the intervention of a network operator when necessary. To summarise, the network generates raw data about its traffic, Monitoring and Detection reduce this to simpler signals, and the remaining functions feed orders back into the network to either defeat an attack or mitigate its effects.

To address the problem of heterogeneity and enable the development of reusable strategies for detection and remediation, we separate domain-agnostic components Detection and Reaction from domain-specific components Monitoring and Remediation. Detection and Reaction together embody a strategy for determining the existence of an attack and a response to it, while Monitoring and Remediation respectively implement how traffic is collected from the network and how strategies are applied.

The trade-off that determines under what circumstances automated remediation should be used is realised through the configuration of Reaction components. Detection components inform Reaction of the existence and severity of perceived malicious behaviour. Reaction then determines a course of action, also depending on available remediation mechanisms. If appropriate, Visualisation components are used to aid a network operator when making decisions about what remediation activities to invoke, by providing details of the current attack and other network state.

3.1 Monitoring and Detection

IDSs use information on network traffic in order to detect ongoing malicious activities. A wide-scope view of the network traffic as well as a deep knowledge of

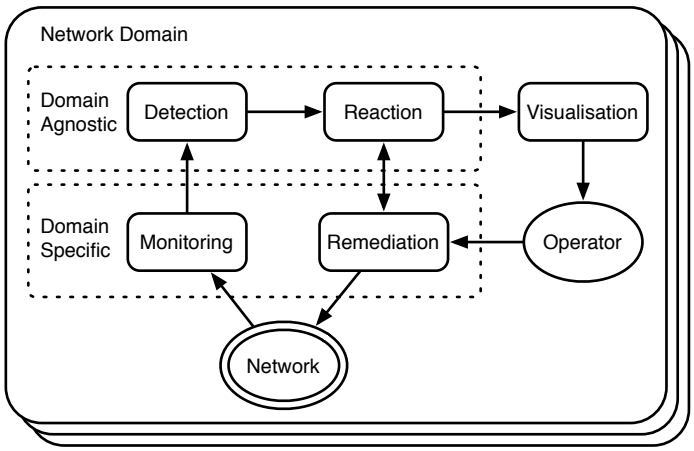

Fig. 1. The INTERSECTION framework

the network status improves the detection process. For example, a Distributed Denial of Service (DDoS) attack is performed by systems that are widely distributed throughout the network. In order to effectively detect such attacks, information regarding a number of active traffic flows and different network entities is required. Such entities share the same "purpose" (i.e., to jeopardize either a service or a single host) and make use of the same resources (e.g., those belonging to the network infrastructure) in order to accomplish their task. They cooperate in order to achieve the same objective. This malicious cooperation can be detected only if a wide-scope analysis of traffic flows is performed.

This kind of analysis requires using coarser grained flow definitions which convey, for example, the traffic from different sources to one destination. Since the amount of resources needed for measurement and reporting increases with the level of granularity required to detect an attack, a multi-step monitoring approach is useful. We call this approach *adaptive monitoring*. An approach to granularity adaptive attack detection is presented in [9].

In the framework, the Monitoring function encompasses all mechanisms for observing traffic in the network and condensing the information about it (e.g., by gathering statistics into flows, or observing simple statistics about the whole network). Detection configures Monitoring to receive data from it, and attempts to interpret that data as anomalous or indicative of an intrusion. As a result, it may simply report that an anomaly is detected, or it may report the degree, plus other parameters such as the end-systems to which it pertains, or the level of confidence it has that a genuine intrusion is taking place. Importantly, it may also reconfigure Monitoring to obtain greater detail temporarily, in order to make a better determination, hence the level of monitoring is adapted according to the need to make more detailed detection decisions.

The choice of Monitoring functions is fixed by the configuration of the network – development here is driven by hardware and associated management software. In contrast, Detection mechanisms may be added to the network dynamically – a programmer works here to improve intrusion tolerance.

3.2 Reaction and Remediation

Reaction represents pre-determined decisions or logic for handling anomalies reported by Detection. Options include reporting the anomaly through IDMEF [8] to other domains, reporting it to a network operator via Visualization, and requesting more information via Detection. Through a chain of detections and reactions, the system may detect more sophisticated intrusions without applying intrusive or resource-hungry network monitoring at all times.

Remediation encompasses all mechanisms provided by the network for defeating an intrusion, or for mitigating its effects. It is an option for Reaction to trigger Remediation having detected a particular kind of intrusion, and might (for example) isolate a compromised node from its peers, or rate-limit its probing attempts. This activity might occur even before an intrusion is fully identified, i.e., at earlier points in the "chain of detections and reactions".

Again, there is a similar distinction between Reaction and Remediation as there is between Monitoring and Detection – the Remediation mechanisms available in a network are fixed, i.e., determined by its configuration, while Reaction mechanisms may be added dynamically.

3.3 Visualization and the Operator

Autonomic remediation will not be enough to deal with the evolution of attacks in the long term. Most forms will involve a certain cost to the network as an alternative to a possible catastrophic cost of network failure, and they may need Operator interaction to determine when to be switched off. The Operator may also be required to deal with new attacks (typically appearing as anomalies) for which a trustworthy autonomic remedy has not yet been devised or deployed. The Visualization function covers the reporting of attacks and anomalies to the Operator, and may receive such signals from Reaction, or additional details from Detection and Monitoring. Visualization together with the Operator could be regarded as a form of remediation – they complete the control loop back to the network with a slow path.

3.4 Recovery

Applying a remedy indefinitely could be costly. Some remedies involve only a one-off cost, but thereafter cost nothing, and could be left applied. Indeed, they could be applied even before any intrusion or anomaly is detected. They consist of patches to fix infrequent bugs and exploits, and changes to newer, more robust protocols.

In contrast, some remedies can only be applied temporarily because they impede the normal operation of some part of the network, and should be withdrawn as soon as possible. These include isolation of compromised nodes (fixed by disinfecting and patching the nodes), and blocking of attack traffic (fixed by detecting the end of an attack) — not eventually withdrawing these remedies renders the affected node's participation in the network pointless. The necessary withdrawal of a remedy to achieve normal operation is referred to as *recovery*.

How should the end of an attack be detected? Two choices are:

1. To re-use Monitoring and Detection, which originally reported the attack, to also report its termination.
2. To get Remediation to perform very anomaly-specific and narrow-scope monitoring at its deployment locations (and perhaps report back to Reaction to make a wide-scope decision to withdraw).

In the first case, there are several options:

Remediation-Monitoring interaction Remediation could inform Monitoring of what it is doing, e.g., which flows it is blocking, and report flows that Monitoring no longer should see.
A problem with this approach is that it couples Monitoring and Remediation. This potentially requires devising a complex expression of what Remediation is doing, in order for Monitoring to understand it, or would make it harder for Monitoring and Remediation implementations to be developed independently.

Monitoring non-interference In its reports to Reaction, Detection could indicate the monitoring sources which precipitated those reports. Reaction could use this to deploy remedies at locations that protect the victim but do not affect Monitoring; Detection then will later signal the end of the attack.

Monitoring migration Reaction informs Detection where it is applying Remediation, and so Detection adjusts the location of Monitoring to be able to continue to detect the attack.
This approach seems impractical. Firstly, the possible locations for Monitoring are likely to be fixed, or if they are good enough when moved, why not simply monitor at the preferred locations anyway? Secondly, two different remedies applied independently at the same time may impose contradictory requirements on Monitoring.

Additionally, all of these approaches do not fit well with recovery that requires a manual trigger (e.g., acknowledging that a compromised node has been cleaned), as it implies that the Operator must also try to convince Monitoring that the threat has passed, instead of simply withdrawing the remedy directly. Furthermore, if Detection is based on traffic from the victim (e.g., anomalous responses to an attack), any successful remedy will surely stop these from happening.

Therefore, we follow the second case, i.e., Remediation should do its own narrow-scope monitoring to detect the end of the attack. This has the following

advantages: it leaves Remediation decoupled from Monitoring; the relative positions of Remediation and Monitoring become irrelevant; and Detection can still be based on the victim's response. However, the decision to withdraw can still be placed with Reaction; this will gather reports from Remediation to make a decision based on the wider view it has of the attack, which individual Remediation deployments do not.

3.5 Example Scenario

To demonstrate how the INTERSECTION framework can be applied, we present an example scenario, which is depicted in Fig. 2. In this scenario, a Web server is being subjected to a Distributed Denial of Service (DDoS) attack. The domain that provides connectivity to the Web server is enabled with mechanisms that implement our framework.

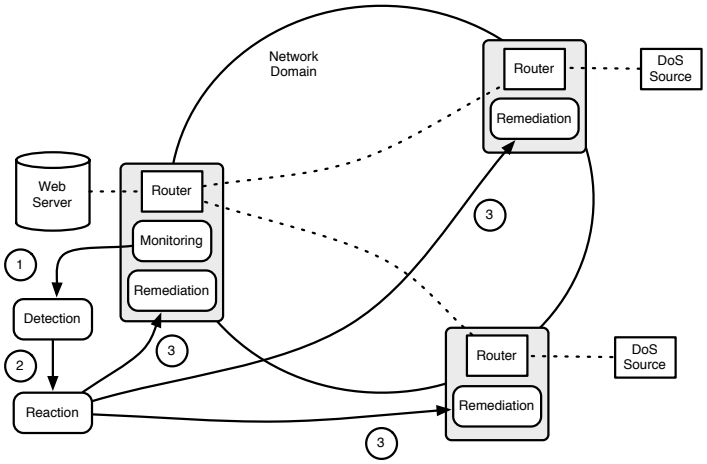

Fig. 2. Use of the INTERSECTION components in an example scenario

The edge router associated with the Web server is able to monitor incoming and outgoing traffic volumes. This summarised information is passed to a detection component, labelled as step one in Fig. 2, which uses this information to detect the onset of problems associated with an attack. Xie *et al.* [15] describe an algorithm that enables the detection of the onset of problems associated with a flash crowd (or similarly, a DDoS attack) on a Web server, which uses a disparity between the expected response traffic volume and the actual traffic seen by monitoring, to suggest either network congestion or system overload. The detection mechanism in this scenario implements this algorithm.

If a problem is detected, Reaction components are informed with details of the attack (e.g., the source addresses of attackers and the severity of the attack,

which in this case relates to the magnitude of the disparity of expected and actual response traffic volumes). Reaction can then initiate some form of remediation, if necessary, or inform the Operator via Visualisation (not shown in Fig. 2). Let us assume this attack is particularly intense and the network provider has a contract to protect the Web server. In this instance, Reaction components initiate a rate-limiting remediation mechanism. The severity of the rate-limiting is dictated by the magnitude of the problem as determined by detection (as Xie *et al.* propose [15]). Rate-limiting is invoked on the edge router associated with the Web server (perhaps, initially) and on the ingress routers to the network providing connectivity to the Web server.

When the rate-limiting is invoked on the ingress routers to the network, the Detection component associated with the edge router will naïvely assume the attack has ended and inform Reaction of the end of the attack. Hence the withdrawal of the remedy has to be determined by information from the remediation components themselves. Here, the rate-limiting at the edge router near the Web server could stop with the invocation of the remedy at the ingress routers.

4 Framework Realisation – Autonomic Distributed IDS

This section describes the design and implementation of Monitoring and Detection components of the INTERSECTION framework. If we assume that the network is aware of itself, security assurance might be regarded as a *service* inherently provided by the network infrastructure. In such a scenario, we can think of a system capable of deploying, both proactively and reactively, on-demand security services.

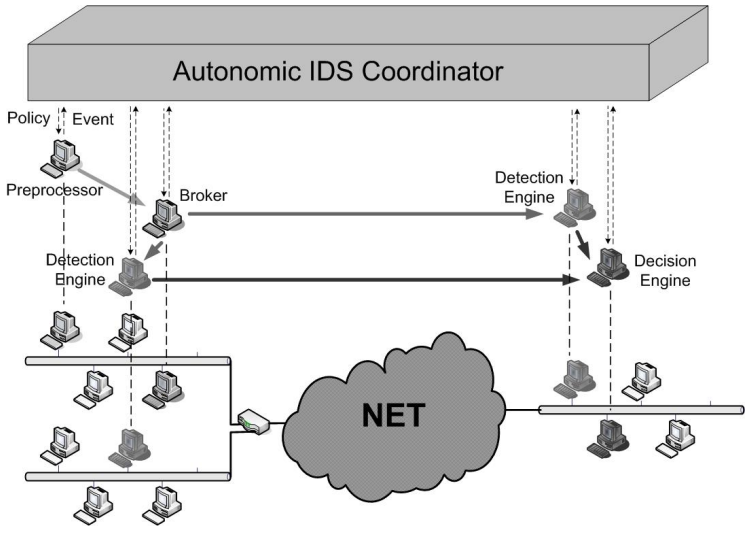

Fig. 3. A model for a distributed autonomic IDS

We propose a system (Fig. 3) which dynamically deploys the entities in charge of providing network security, based on knowledge of the *security status* of the network and its components. The *Preprocessor* and *Broker* entities implement the Monitoring function, collectively the *Detection Engine* and *Decision Engine* entities the Detection function, and the *Autonomic Co-ordinator* implements the Reaction functionality. These entities are described in more detail below.

- *Preprocessor*
  Each such component is in charge of:
  1. capturing raw data from the network;
  2. extracting information (e.g. traffic features, flow information, etc.) from captured packets;
  3. sending the computed information to a *broker* entity.
- *Broker*
  The broker is a mediation component whose main tasks are:
  1. collecting information from all of the preprocessors spread across the network;
  2. distributing the collected information to one or more *detection engines*. The distribution strategy is defined by means of a policy-based approach.
- *Detection Engine*
  Detection engines receive traffic information from the broker and decide on whether or not such information represents a potential attack pattern, based on a specific detection technique.
- *Decision Engine*
  The main task of the decision engine resides in collecting information coming from detection engines and coming up with a final decision concerning the current network context. This might be done through a number of approaches, ranging from simple majority voting to much more complex solutions in which the various inputs provided by the detection engines are appropriately weighted on the basis of their degree of reliability (with respect to their own capability of detecting a particular set of malicious activities). Interaction between each detection engine and the decision engine might be realised using the IDMEF (Intrusion Detection Message Exchange Format) standard data format [8].
- *Autonomic Coordinator*
  This entity is responsible for the coordination of the other system components. It will work on the basis of a notification paradigm: the various system components can be seen as information producers, whereas the coordinator itself acts as an information consumer. Each time something worth reporting happens, an event is generated and captured by the coordinator which triggers the appropriate configuration task, e.g. the invocation of some form of Remediation activity.

In order to improve the system's performance and reliability, a diversity-based approach to the selection of detection techniques might be successfully exploited. In fact, by integrating different mechanisms for attack classification,

the system may increase its detection capabilities, since more accurate analysis can be performed in order to detect specific attacks. Support for reliability evaluation can be added when multiple detection techniques are used, thus allowing evaluation of the accuracy of each issued decision. A multi-classification approach can be implemented either in a centralized or a distributed fashion: the combination of multiple detection techniques can be used both locally, in order to increase the reliability of alerts raised at each detection engine, and in a distributed fashion, by combining evidence of events communicated by different detection engines to the decision engine, in order to gain a global knowledge of the overall security status of the monitored networking scenario.

Dynamic deployment of the distributed system components is needed in order to ensure both flexibility of the architecture and robustness in the face of changes in network and traffic conditions. As an example, the IDS might need several preprocessors sniffing traffic in crucial points of the network, as well as several detection engines, each exploiting the best fitting detection technique according to the system status and node location. Such a dynamically distributed system might, for instance, adapt itself at the occurrence of a DDoS attack, by appropriately placing IDS engines in the most critical nodes (i.e., the nodes along the attackers' path) and by coordinating such nodes through a proper protocol (e.g., a protocol for tracing back the attack).

5 Conclusion

We have presented a framework for deploying network intrusion-tolerance systems, taking into consideration: the cost of performing over-detailed, full-time monitoring by adapting the level of monitoring according to current suspicions; the need to deal with different kinds of intrusions and deploy different remedies according to the available facilities of the network; the need to leave some decisions to manual operation; the need to recover the network automatically.

We have shown how the framework is congruent with the implementation plans described in Section 4. To date, a prototype implementation of the distributed IDS is being tested. For the autonomic coordinator, we have adopted an approach based on the Grid services paradigm: it is based on the WS-GRAM (Web Service Grid Resource Allocation and Management) module of the Globus Toolkit[4] and takes care of dynamically deploying the various system components.

Further work will involve investigating the suitability of our chosen approach to recovery, i.e., not to work around the monitoring used to detect intrusions, but to perform narrow-scope monitoring as part of remediation.

Acknowledgements

This research is funded by the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216585 (INTERSECTION Project).

[4] See `http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WSGRAMFacts.html`

References

1. The INTERSECTION Project. http://www.intersection-project.eu/.
2. Cristina Abad, Yifan Li, Kiran Lakkaraju, Xiaoxin Yin, and William Yurcik. Correlation between netflow system and network views for intrusion detection.
3. Cristina Abad, Jed Taylor, Cigdem Sengul, William Yurcik, Yuanyuan Zhou, and Ken Rowe. Log correlation for intrusion detection: A proof of concept. In *Proceedings of the 19th Annual Computer Security Applications Conference, (ACSAC)*, 2003.
4. D. Armannsson, P. Smith, G. Hjalmtysson, and L. Mathy. Controlling the Effects of Anomalous ARP Behaviour on Ethernet Networks. In *CoNEXT 2005*, pages 50–60, Toulouse, France, October 2005.
5. Rebecca Gurley Bace. *Intrusion Detection*. Macmillan Technical Publishing, January 2000.
6. Andrew R. Baker, Brian Caswell, and Mike Poor. *Snort 2.1 Intrusion Detection - Second Edition*. Syngress, 2004.
7. I. Balepin, S. Maltsev, J. Rowe, and K. Levitt. Using Specification-Based Intrusion Detection for Automated Response. In *International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*, 2003.
8. H. Debar, D. Curry, and B. Feinstein. The Intrusion Detection Message Exchange Format (IDMEF). Number 4765 in RFC. IETF, March 2007.
9. Thomas Gamer, Marcus Schöller, and Roland Bless. A granularity-adaptive system for in-network attack detection. In *IEEE / IST Workshop on Monitoring, Attack Detection and Mitigation*, pages 47–50, Tuebingen, Germany, September 2006.
10. Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *Proceedings of ACM SIGCOMM '05*, August 2005.
11. Zhenmin Li, Jed Taylor, Elizabeth Partridge, Yuanyuan Zhou, William Yurcik, Cristina Abad, James J. Barlow, and Jeff Rosendale. Uclog: A unified, correlated logging architecture for intrusion detection. In *Proceedings of the 12th International Conference on Telecommunication Systems - Modeling and Analysis (ICTSM)*, 2004.
12. Ulf Lindqvist and Phillip A Porras. Detecting computer and network misuse through the production-based expert system toolset (p-best). In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pages 146–161, Oakland, California, may 1999. IEEE Computer Society Press, Los Alamitos, California.
13. Matthew Vincent Mahoney. Network traffic anomaly detection based on packet bytes. In *Proceedings of ACM SAC 03*, 2003.
14. Vern Paxson and Brian Terney. Bro reference manual, 2004.
15. L. Xie, P. Smith, A. Jabbar, M. Banfield, H. Leopold, D. Hutchison, and J. P.G. Sterbenz. From Detection to Remediation: A Self-Organized System for Addressing Flash Crowd Problems. In *IEEE International Conference on Communications (ICC 2008)*, Beijing, China, May 2008.
16. Xiaoxin Yin, William Yurcik, Michael Treaster, Yifan Li, and Kiran Lakkaraju. Visflowconnect: netflow visualizations of link relationships for security situational awareness. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 26 – 34. ACM Press, 2004.
17. Y. Zhang, J. Luo, and H. Lu, editors. *Wireless Mesh Networking Architecture, Protocols and Standards*. Auerbach Publications, 2007.